

16.35

Aerospace Software Engineering

Ada 95 – “some basics”



September 16/2002
Prof. I. K. Lundqvist
kristina@mit.edu

Ada ...

- Cost study done in 1973-1974 determined that the US Department of Defense was spending \$3 billion annually on software
- Evaluated 23 existing languages against the “Tinman” requirements:
 - FORTRAN, COBOL, **PL/I**, HAL/S, TACPOL, CMS-2, CS-4, SPL/I, JOVIAL J3, JOVIAL J73, ALGOL 60, **ALGOL 68**, CORAL 66, **Pascal**, SUMULA 67, LIS, LTR, TRL/2, EUCLID, PDL2, PEARL, MORAL, EL/I
- Augusta **Ada** Byron (1815-1852), Countess of Lovelace

A Small Ada Program - Mini

```
-- What is program supposed to be doing? Inputs/outputs?  
-- Author(s):  
-- Last updated:  
-- Comments:  
  
with Ada.Text_IO;  
  
procedure Mini is  
    -- declarations  
    K      : constant Integer := 5;  
    X, Y   : Integer;  
  
begin  
    -- executable code  
    Ada.Text_IO.Put (Item => "Hello world!");  
    X := 10;  
    Y := X * K; -- calculations  
    Y := X + 1;  
end Mini;
```

Reserved Words

- These identifiers are reserved for special significance in the language
- A *reserved word* must not be used as a declared identifier

<code>abort</code>	<code>else</code>	<code>new</code>	<code>select</code>
<code>abs</code>	<code>elsif</code>	<code>not</code>	<code>separate</code>
<code>abstract</code>	<code>end</code>	<code>null</code>	<code>subtype</code>
<code>accept</code>	<code>entry</code>		
<code>access</code>	<code>exception</code>	<code>of</code>	<code>tagged</code>
<code>aliased</code>	<code>exit</code>	<code>or</code>	<code>task</code>
<code>all</code>		<code>others</code>	<code>terminate</code>
<code>and</code>	<code>for</code>	<code>out</code>	<code>then</code>
<code>array</code>	<code>function</code>		<code>type</code>
<code>at</code>		<code>package</code>	
	<code>generic</code>	<code>pragma</code>	<code>until</code>
<code>begin</code>	<code>goto</code>	<code>private</code>	<code>use</code>
<code>body</code>		<code>procedure</code>	
	<code>if</code>	<code>protected</code>	<code>when</code>
<code>case</code>	<code>in</code>		<code>while</code>
<code>constant</code>	<code>is</code>	<code>raise</code>	<code>with</code>
		<code>range</code>	
<code>declare</code>	<code>limited</code>	<code>record</code>	<code>xor</code>
<code>delay</code>	<code>loop</code>	<code>rem</code>	
<code>delta</code>		<code>renames</code>	
<code>digits</code>	<code>mod</code>	<code>requeue</code>	
<code>do</code>		<code>return</code>	
		<code>reverse</code>	

Finding the Value of a Coin Collection

- Problem Specification

- Your little sister has been saving nickels and pennies for quite a while. Because she is getting tired of lugging her piggy bank with her whenever she goes to the store, she would like to trade in her collection for one-dollar banknotes and some change. To do this, she would like to know the value of her coin collection in dollars and cents.

- Analysis

- To solve this problem, we must be given the count of nickels and pennies in the collection. The first step is to determine the total value of the collection in cents. Once we have this figure, we can do an integer division using 100 as the divisor to get the dollar value; the remainder of this division will be the loose change that she should receive. In the data requirements below, we list the total value in cents (`TotalCents`) as a program variable because it is needed as part of the computation process; it is not a required problem output.

Finding the Value of a Coin Collection

- Data Requirements and Formulas
 - Problem Inputs:
 - Nickels : Natural (the number of nickels)
 - Pennies : Natural (the number of pennies)
 - Problem Outputs:
 - Dollars : Integer (the number of \$s she should receive)
 - Change : Integer (the loose change she should receive)
 - Additional Program Variables:
 - TotalCents : Integer (the total number of cents)
 - Relevant Formulas
 - One nickel equals 5 pennies

Finding the Value of a Coin Collection

- Design
 - Initial Algorithm
 1. Read in the count of nickels and pennies
 2. Compute the total value in cents
 3. Find the value in dollars and loose change
 4. Display the value in dollars and loose change

Finding the Value of a Coin Collection

- Design
 - Refined Algorithm
 1. Read in the count of nickels and pennies
 2. Compute the total value in cents
 1. `TotalCents` is 5 times `Nickels` plus `Pennies`
 3. Find the value in dollars and loose change
 1. `Dollars` is the integer quotient of `TotalCents` and 100
 2. `Change` is the integer remainder of `TotalCents` and 100
 4. Display the value in dollars and loose change

Finding the Value of a Coin Collection

- Test Plan

Test case	Nickles	Pennies	Reason	Expected output
1	30	77	Typical	\$2.27
2	0	59	No nickels	\$0.59
3	13	0	No pennies	\$0.65
4	13	-5	Negative	?
5	0	0	No coins	\$0.00
6	qwerty	4	Bad input	?

Finding the Value of a Coin Collection

```
with Ada.Text_IO;
with Ada.Integer_text_IO;
procedure Coin_Collection is
  Pennies      : Natural; -- input, number of pennies
  Nickels      : Natural; -- input, number of nickels
  Dollars      : Natural; -- output, value in dollars
  Cents        : Natural; -- output, value in cents
  Totalcents   : Natural;

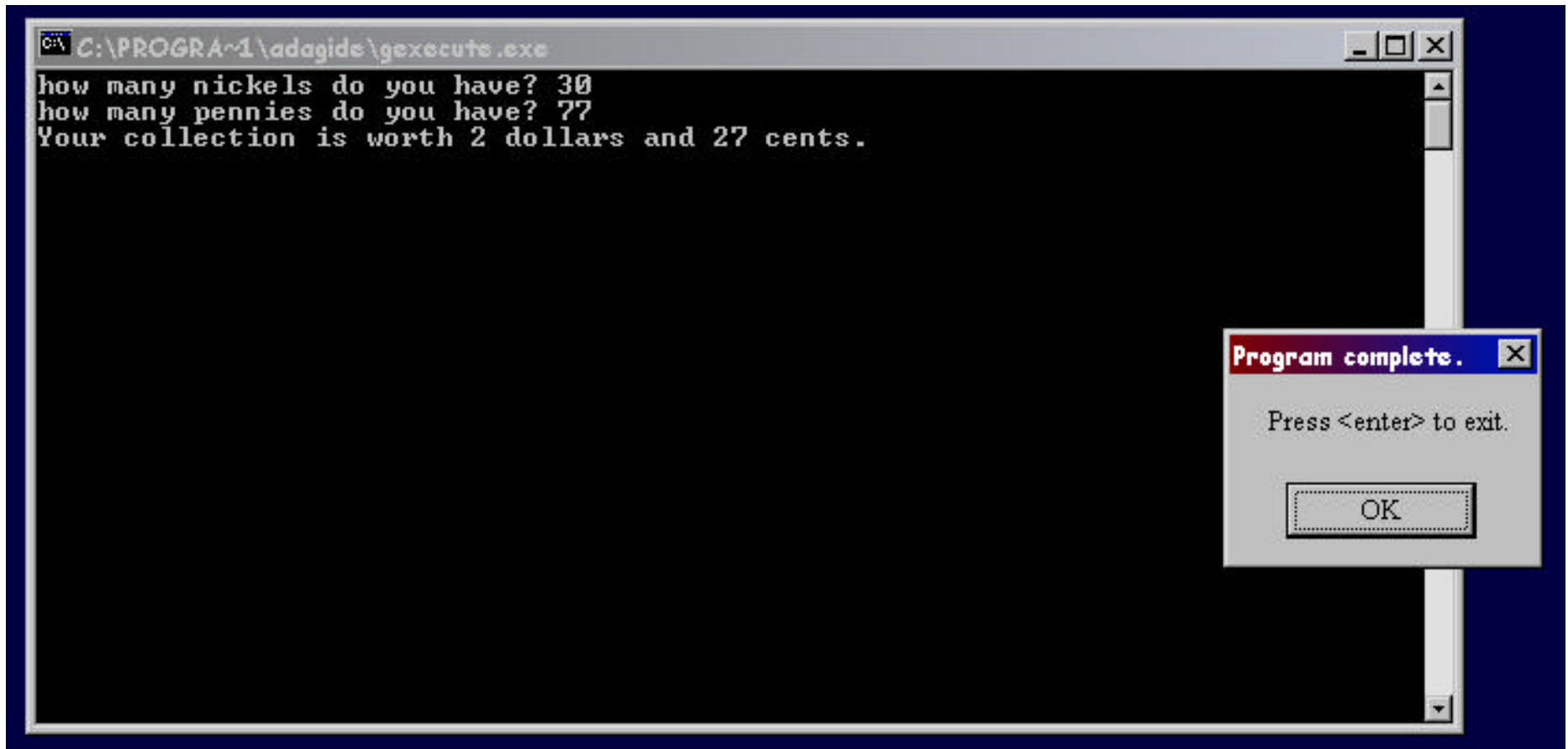
begin -- coin_collection
  --prompt user for number of nickels and pennies
  Ada.Text_IO.Put (Item => "how many nickels do you have? ");
  Ada.Integer_Text_IO.Get (Item => Nickels);
  Ada.Text_IO.Put (Item => "how many pennies do you have? ");
  Ada.Integer_Text_IO.Get (Item => Pennies);

  Totalcents := 5 * Nickels + Pennies; -- compute total value in cents

  Dollars := Totalcents / 100; -- find value in dollars and change
  Cents := Totalcents rem 100;

  -- display the value in dollars and change
  Ada.Text_IO.Put (Item => "Your collection is worth ");
  Ada.Integer_Text_IO.Put (
    Item => Dollars,
    Width => 1);
  Ada.Text_IO.Put (Item => " dollars and ");
  Ada.Integer_Text_IO.Put (
    Item => Cents,
    Width => 1);
  Ada.Text_IO.Put (" cents.");
  Ada.Text_IO.New_Line;
end Coin_Collection;
```

Testing



The image shows a Windows command prompt window titled "C:\PROGRAM~1\adagide\gexecute.exe". The prompt displays the following text:

```
how many nickels do you have? 30  
how many pennies do you have? 77  
Your collection is worth 2 dollars and 27 cents.
```

An "OK" dialog box is overlaid on the bottom right of the command prompt. The dialog box has a title bar that says "Program complete." and contains the text "Press <enter> to exit." with an "OK" button below it.

Common Programming Errors

- Compilation errors, run-time errors (*exceptions* in Ada), and logic or algorithm errors.
- Compilation errors:
 - Distance_with_Errors
 - Finds distance traveled, given travel time and average speed

Distance_With_Errors

```
with Ada.Text_Io;
with Ada.Float_Text_Io;
procedure Distance_With_Errors is
  -- Finds distance, given travel time and average speed
  Howlong : Natural;
  Howfast : Float;
  Howfar : Natural;

begin -- distance_with_errors
  -- prompt user for hours and average speed
  Ada.Text_Io.Put (Item => "How long will you be driving (integer) ? ");
  Ada.Float_Text_Io.Get (Item => Howlong);
  Ada.Text_Io.Put (Item => "At what speed (miles per hour, integer) ?");
  Ada.Float_Text_Io.Get (Item => Howfast);

  -- compute distance driven
  Howfast := Howlong * Howfar;

  -- display result
  Ada.Text_Io.Put (Item => " you will travel about ");
  Ada.Float_Text_Io.Put (Item => Howfar);
  Ada.Text-Io.Put (Item => " miles");
  Ada.Text_Io.New_Line;

end Distance_With_Errors;
```

Distance_With_Errors

GNAT 3.14p (20010503) Copyright 1992-2001 Free Software Foundation, Inc.

Compiling: c:\gnat\lib\gcc-lib\pentiu~1\282a9d~1.1\adainc~1\howlong);adb (source file time stamp: 2002-09-15 20:28:54)

```
1. with Ada.Text_Io;
2. with Ada.Float_Text_Io;
3. procedure Distance_With_Errors is
4.   -- Finds distance, given travel time and average speed
5.   Howlong : Natural;
6.   Howfast : Float;
7.   Howfar  : Natural;
9. begin -- distance_with_errors
10.  -- prompt user for hours and average speed
11.  Ada.Text_Io.Put (Item => "How long will you be driving (integer) ? ");
12.  Ada.Float_Text_Io.Get (Item => Howlong);
13.  Ada.Text_Io.Put (Item => "At what speed (miles per hour, integer) ?");
14.  Ada.Float_Text_Io.Get (Item => Howfast);
16.  -- compute distance driven
17.  Howfast := Howlong * Howfar;
19.  -- display result
20.  Ada.Text_Io.Put (Item => " you will travel about ");
21.  Ada.Float_Text_Io.Put (Item => Howfar);
22.  Ada.Text-Io.Put (Item => " miles");
    |
    >>> missing ";"
23.  Ada.Text_Io.New_Line;
24.
25. end Distance_With_Errors;
25 lines: 1 error
```

Distance_With_Errors

```
1. with Ada.Text_Io;
2. with Ada.Float_Text_Io;
3. procedure Distance_With_Errors is
4.
5.   Howlong : Natural;
6.   Howfast : Float;
7.   Howfar  : Natural;
8.
9. begin -- distance_with_errors
10.  -- prompt user for hours and average speed
11.  Ada.Text_Io.Put (Item => "How long will you be driving (integer) ? ");
12.  Ada.Float_Text_Io.Get (Item => Howlong);
      |
      >>> invalid parameter list in call
13.  Ada.Text_Io.Put (Item => "At what speed (miles per hour, integer) ?");
14.  Ada.Float_Text_Io.Get (Item => Howfast);
15.
16.  -- compute distance driven
17.  Howfast := Howlong * Howfar;
      |
      >>> expected type "Standard.Float"
      >>> found type "Standard.Integer"
18.
19.  -- display result
20.  Ada.Text_Io.Put (Item => " you will travel about ");
21.  Ada.Float_Text_Io.Put (Item => Howfar);
      |
      >>> invalid parameter list in call
      >>> possible missing instantiation of Text_IO.Integer_IO
22.  Ada.Text_Io.Put (Item => " miles");
23.  Ada.Text_Io.New_Line;
24.
25. end Distance_With_Errors;
25 lines: 5 errors
```

Finding the Value of a Coin Collection

- Exceptions (Run-Time Errors)
 - How many nickels do you have? 13
How many pennies do you have? -5
 - CONSTRAINT ERROR
 - Variables out of range in their programs
 - How many nickels do you have? qwerty
 - Ada.IO_EXCEPTIONS.DATA_ERROR
 - Input/output exception

Control Structures

- So far: straight-line algorithms
- Control structures: `if`, `for`, `while`

```
if X >= 0.0 then
    Ada.Text_Io.Put (Item => "Positive");
else
    Ada.Text_Io.Put (Item => "Negative");
end if;
```

```
if X > 0.0 then
    Posprod = Posprod * X;
    Countpos = Countpos + 1;
end if;
```

For

```
Ada.Text_Io.Put(Item => "Hello there. ");  
Ada.Text_Io.Put(Item => "Hello there. ");  
Ada.Text_Io.Put(Item => "Hello there. ");  
Ada.Text_Io.Put(Item => "Hello there. ");  
Ada.Text_Io.Put(Item => "Hello there. ");
```

--Can Be Written More Concisely As

```
for Count in 1..5 loop  
    Ada.Text_Io.Put(Item => "Hello there. ");  
end loop;
```

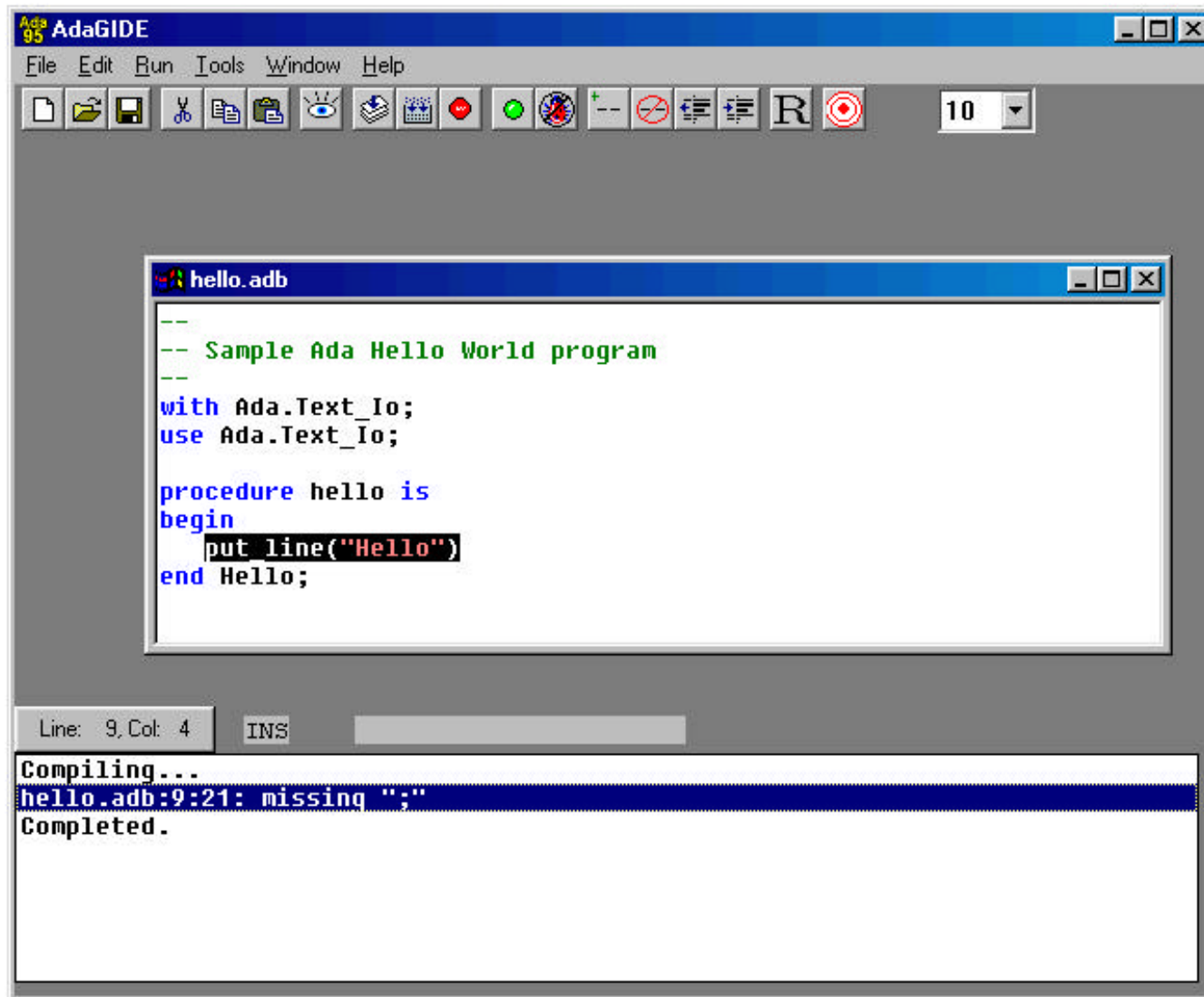
```
for Counter in Min .. Max loop  
    Ada.Integer_Text_Io.Put (  
        Item => I,  
        Width => 5);  
    Ada.Text_Io.New_Line;  
end loop;
```

While

```
Power := 1;  
while Power < 10000 loop  
  Ada.Integer_Text_Io.Put (  
    Item => Power,  
    Width => 5);  
  Power = Power * 2;  
end loop;
```

```
Power := 1;  
loop  
  exit when Power >= 10000;  
  Ada.Integer_Text_Io.Put (  
    Item => Power,  
    Width => 5);  
  Power = Power * 2;  
end loop;
```

AdaGIDE



The screenshot shows the AdaGIDE IDE interface. The main window displays the source code for a file named `hello.adb`. The code is as follows:

```
--  
-- Sample Ada Hello World program  
--  
with Ada.Text_IO;  
use Ada.Text_IO;  
  
procedure hello is  
begin  
  put_line("Hello")  
end Hello;
```

The cursor is positioned at the end of the `put_line("Hello")` line. Below the code editor, the status bar shows "Line: 9, Col: 4" and "INS". At the bottom of the IDE, a console window displays the following output:

```
Compiling...  
hello.adb:9:21: missing ";"  
Completed.
```

The GNU Visual Debugger

The screenshot shows the GNU Visual Debugger (GDB) interface. The window title is "The GNU Visual Debugger". The menu bar includes "File", "Edit", "Program", "Command", "Data", and "Help". The toolbar contains icons for "run", "start", "step", "stepi", "next", "nexti", "finish", "cont", "up", "down", and "interrupt".

Annotations with orange arrows point to various parts of the interface:

- Toolbar**: Points to the top toolbar.
- Canvas**: Points to the central graph area showing memory addresses and values.
- Highlighted Source Explorer**: Points to the left sidebar showing a tree view of source files, with "parse.adb" highlighted.
- Breakpoint Debugger Window**: Points to the bottom status bar showing breakpoint information.
- Dereference**: Points to a graph node labeled "5: A".
- Record type**: Points to a graph node labeled "7: W" which displays a record structure with fields.
- Unknown value**: Points to a graph node labeled "12: Tcb".
- Current line**: Points to line 110 in the source code window, which is highlighted in green.

The source code window displays the following code:

```
90
91 type String_Access is access String;
92 type Array_of_String is array (1 .. 2) of String_Ace +
93 Aoa : Array_of_String := (new String' ("ab"), new Stri +
94
95 type Null_Record is null record;
96 Nr : Null_Record;
97
98 type My_Record is record
99   Field1 : Access_Type;
100   Field2 : String (1 .. 2);
101 end record;
102 V : aliased My_Record := (Field1 => A'Access, Field2 =>
103
104 type Complex_Repeat_Type is array (1 .. 100) of My_Re +
105 Crt : Complex_Repeat_Type := (others => V);
106
107 type My_Record_Access is access all My_Record;
108 Mra : My_Record_Access := V'Access;
109
110 type My_Record_Array is array (1 .. 100) of My_Re +
111 W : My_Record_Array := ((Field1 => A'Access, Field2 =>
112   (Field1 => A'Access, Field2 =>
```

- Problem set 1 due 9/25/02, 3pm
- Email addresses:
 - 16.35-students@mit.edu
 - 16.35-staff@mit.edu
- <http://web.mit.edu/16.35/www>
 - Class project
 - [X-38 Integrated Test Plan](#)
 - [X-38 Software Development Plan](#)
 - [Early Flight Control System - Statement of Work](#)
 - [Software Development Plan](#)
 - [Software Test Plan](#)

Project Groups

- Mandic, Milan
Nyenke, Chinwe P
da Silva, Lanya M
Riedel, Robin
- Modisette, James M
Guzman, Bryan J
Utter, Darlene A
Bly, Elizabeth
Kambouchev, Nayden
- Phifer, Gabriel S
Sidelnik, Nicholas
Ouellette, Joshua T
Chang, Catherine W
Stringfellow, Margaret
- Minogue, Kenneth
Broniatowski, David
Qu, Shen
Guevara, Gerardo